```python
import random
from datetime import datetime, timedelta
import pandas as pd
from dateutil.parser import parse


class CreatingDB:
    """
    Class for creating random database
    """
    num_people = 0  # number of people to create
    base_date = None  # the base date of data

    def __init__(self, num_people, base_date):
        self.num_people = num_people
        self.base_date = base_date

    def generate_incurred_date(self):
        """
        function to create random incurred date
        :return:
            incurred_date: string, the day of infection or contact
            elapsed_days: int, the difference between base date and incurred date
        """
        elapsed_days = random.randint(0, 14)  # the valid day period is 0~14
        # extracting the incurred day using periods and base date
        incurred_date = (self.base_date - timedelta(days=elapsed_days)).\
            strftime("%Y %m %d")
        return incurred_date, elapsed_days

    def compute_severity(self, status, elapsed_days):
        """
        function to compute severity
        :param status: string, 'contacted' or 'confirmed'
        :param elapsed_days: the period form the incurred date
        :return: the computed severity using formula
        """
        if status == 'contacted':  # contacted person?
            return 1 - (elapsed_days * 0.05)
        else:  # confirmed person?
            return (1 - (elapsed_days * 0.05)) * 0.5

    def generate_address_list(self):
        """
        function to get one address randomly from the adress list
        :return: the randomly generated address list
        """
        with open('./Address_Part.txt', 'r', encoding='utf-8') as add_file:
            # add_file = add_file.encoding
            address_list = add_file.readlines()

            random_address_list = []  # list to store addresses

            # extract addresses as many as the number of recipients
            for _ in range(1, self.num_people+1):
                random_address_list.append(random.choice(address_list))

        return random_address_list
```

```
    def generate_csv_data(self):
        """
        function to create .csv file with randomly generated records
        :return: None
        """
        num_healthy = round(self.num_people / 3)  # 1/3 is healthy
        num_contacted = round(self.num_people / 3)  # 1/3 is contacted
        # 1/3 is confirmed
        num_confirmed = self.num_people - num_healthy - num_contacted

        id_list = list(range(1, self.num_people+1))  # ID as many as people
        random.shuffle(id_list)  # shuffle list

        # age records as many as people
        age_list = list(random.randint(1, 100)
                        for _ in range(1, self.num_people+1))
        # address records as many as people
        address_list = self.generate_address_list()

        severity_list = []  # severity records as many as people
        incurred_date_list = []  # incurred date list including 'None'(healthy)
        status_list = []  # status(Healthy, Contacted, and Confirmed) list

        # Entire people num = healthy + contacted + confirmed
        # Repeat as many healthy people
        for _ in range(num_healthy):
            severity_list.append(0)
            status_list.append('Healthy')
            incurred_date_list.append('None')

        # Repeat as many contacted people
        for count in range(num_contacted):
            date, days = self.generate_incurred_date()
            status_list.append('Contacted')
            severity_list.append(round(self.compute_severity('contacted',  days),
2))
            incurred_date_list.append(date)

        # Repeat as many confirmed people
        for _ in range(num_confirmed):
            date, days = self.generate_incurred_date()
            status_list.append('Confirmed')
            severity_list.append(round(self.compute_severity('confirmed',  days),
2))
            incurred_date_list.append(date)

        # converting as pandas DataFrame data type to save .csv
        df = pd.DataFrame({
            "ID": id_list,
            "Age": age_list,
            "Address": address_list,
            "Covid Status": status_list,
            # "Severity": severity_list,
            "Incurred Date": incurred_date_list,
        })
        df = df.sort_values(['ID'], ascending=[True])
        df.reset_index(drop=True, inplace=True)
```

```
        # saving as .csv file
        df.to_csv("corona_data.csv", mode='w', encoding='utf-8-sig')


if __name__ == '__main__':
    # require the number of people and base date
    num_people = int(input("Enter the number of people: "))
    date_input = input("Enter the base date(Year-Month-Day): ")
    if date_input == '':
        print("The base date is set as today.")
        date = datetime.now().date()
    else:
        date = parse(date_input).date()

    cdb = CreatingDB(num_people, date)  # creating instance
    cdb.generate_csv_data()  # creating .csv file
```